Chapter 2

Preface

This is a book about planning software projects. We are writing it mostly to project managers—those who have to plan and track the correspondence of the planning with reality. We also are writing it to programmers and customers, who also have a vital role to play in planning and developing software.

Planning is not about predicting the future. When you make a plan for developing a piece of software, development is not going to go like that. Not ever. Your customers wouldn't even be happy if it did, because by the time software gets there, the customers don't want what was planned, they want something different.

Eisenhower is credited with saying, "Plans are useless. Planning is vital." We agree. That's why this isn't a book about plans, it's about planning. And planning is so valuable and important, so vital, that it deserves to go on a little every day, as long as development lasts. If you follow the advice in this book, you are going to have a new problem to solve every day—planning—but we won't apologize for that, because without planning, software development inevitably goes off the rails.

This isn't a book about the whole of project management. We don't cover typical project manager jobs such as personnel evaluation, recruiting, and budgeting. We have stuck to the parts of the process we know—getting everybody on the team pointed in one direction, discovering when this is no longer true, and restoring harmony.

Extreme Programming (XP) is the programming discipline (you can use the 'm' word if you want to, we'd rather not, thank you) we are evolving to address the problems of quickly delivering quality software, and then evolving it to meet changing business needs. XP isn't just about planning. It covers all aspects of small team software develop-

ment—design, testing, implementation, deployment, maintenance. However, planning is a key piece of the XP puzzle. (For more about XP, read "Extreme Programming Explained: Embrace Change".)

XP develops long projects by breaking them into a sequence of self-contained 1-3 week projects. During each mini-project (iteration),

- Customers pick the features to be added.
- Programmers add the features.
- Programmers and customers write and maintain automated tests to demonstrate the presence of these features.
- Programmers evolve the design of the system to gracefully support the features.
- Programmers finish the features so they are completely ready to be deployed.

Without careful planning, the process falls apart. The team must choose the best possible features to implement. The team must react as positively as possible to the inevitable setbacks. The team must not overcommit, or they will slow down. The team must not under commit, or the customer won't get value for their money. The job of the daily planner is to help keep the team on track in all these areas.

We come by our project planning ideas by necessity. As consultants, we typically see projects that are mostly dead. They typically aren't doing any planning, or they are doing too much of the wrong sort. We invented these planning techniques as the least project planning that could possibly help in these situations. You will have to take what's here and extend and adapt it to your situation. But if you have no planning, or planning that is strangling your project, what's here works for us.

I have a cunning plan.

Baldrick